Express Mail Label No. US. EG545977090US

Graphical Data Collection Interface

FIELD OF INVENTION

The present invention relates to a graphical user data collection/presentation interface. The invention is particularly suited for use with computer programs intended as personal information assistants, consumer preference/opinion gathering tools, and in similar environments.

COPYRIGHT NOTICE

BACKGROUND OF INVENTION

User interfaces for computer programs and operating systems are well-known in the art. At first, such interfaces were entirely text based, and thus primitive, difficult to use, and lacking in functionality. This limitation, too, restricted their use primarily to a small segment of the population consisting of advanced skill computer users. With the advent of graphical interfaces by Xerox, Apple, Microsoft, etc., the use of computers has expanded dramatically to touch upon the entire potential consuming public. Furthermore, the use of graphical interfaces has improved the functionality of computers significantly, so that functions that once required numerous lengthy text based input parameters can now be performed by simple iconic replacements. For example, the task of copying a file from one system drive to another once required extensive knowledge of the exact syntax format of the system, combined with lengthy keyboard data entry. Now, such function can be performed simply by clicking a mouse button on a graphical representation of such file on the first drive, and then manually dragging and dropping such file onto another icon representing the second drive. Utility, productivity, etc., have all increased substantially now because tasks that once required numerous cumbersome operations can

now be performed in a fraction of the time, and without lengthy operator training procedures. This is due, in part, to the fact that most users can intuitively grasp the nature of a function when it is presented in visual form to mimic a real-life manual operation; in this case, the act of moving a file (represented in icon form to resemble a paper document) from one location (represented in icon form as a filing cabinet) to another (represented in icon form as a different filing cabinet). For a useful background reference on this topic, a reference by Schneiderman, entitled "Designing the User Interface," is recommended.

To date, nonetheless, graphical interfaces have been used in computer applications (those programs running under the operating system) primarily for processing only objective data items. For purposes of the present disclosure, a distinction is loosely made between data items that are *objective* - i.e., can be quantified by some known benchmark outside the user's mental/cognitive impressions - and *subjective*, i.e., those data items that are primarily based on the user's personal preferences. As an example, an objective data item might be the temperature at a particular location and time; this can be represented by a data value - i.e., some number in Centigrade - that can be identified/verified by another measurement tool. Similarly, the amount of money in one's bank account can be quantified numerically with a known denomination. In contemporary programming form, this collection of data from the user shows up in, among other places, personal financial planning programs, which might ask a user to identify the real rate of return (a % number) expected/desired by the user for his/her investments.

In contrast, a subjective data item could be the personal enjoyment an individual attains as a result of listening a particular piece of music, the importance they assign to one factor that is part of particular choice they make, etc. For instance, a typical person purchasing an automobile could rank the following in the order of importance in their selection of a particular model: price, performance, comfort, insurance, and so on. Similarly, when asked why a specific course of action was taken, such as selling an equity position in a company, an individual might identify that he/she thought the price had

reached a subjective target value, or that the company was lagging its competitors, or that the local newspaper ran a negative article on the company, etc., etc.

It should be understood that these are but typical examples, and it is apparent that a multitude of similar situations arise each day in connection with a person's experiences/interactions with the world around them. The above, of course, is a very limited explanation of the general differences between subjective and objective data items, and, of course, should be understood as such. There may be a relationship between the two, as for example when an objective data item (the occurrence of an event such as the announcement of a new product by a company) affects and/or results in a subjective data item (a person deciding to buy the company's stock). Thus, this classification is intended merely as an illustrative tool for explaining the basis of the present invention, and should not be taken as a limitation on any of the teachings set forth herein.

Some typical reasons why contemporary programs do not handle such subjective information, include, of course, the fact that few such programs are able to translate this information into machine-manipulable form so that meaningful conclusions can be drawn from the same, and that such results can be conveyed in some intelligent fashion to the user. Fewer still are able to collect this data in an efficient, user-friendly manner; those that do collect subjective data items do so using relatively primitive techniques. For example, the same personal financial planning program described above might ask a user to identify the level of risk he/she is willing to accept, by placing an electronic check mark on a screen form listing such options as "High," "Medium," "Low," etc. Similarly, a conventional on-line purchasing/marketing form might ask the user to identify on a scale of 1 - 10 the importance of various features of a product. To receive the user's input, an electronic representation of a sliding scale might be presented to the user, which he/she can manipulate and set with a computer mouse to a particular value. This is one means of effectuating the graphical object - action interface described above, but is somewhat limited because the user is required to adopt and accept the graphical objects, tools, and options presented to express his/her choices.

The general problems associated with interfaces that attempt to extract individual

subjective data items include the fact that: (1) they rely on non-graphical interfaces, which make them unattractive, unintuitive and unfriendly to prospective users; (2) they present the user with a limited set of objects and interactions for his/her use; in other words, an online questionnaire for example might ask only about the four most important variables as perceived by the vendor, when in fact there may be a whole slew of factors important to the potential customer filling out such questionnaire; (3) they do not permit a user to ignore those items that are not of interest in the provided for universe of parameters; instead, they require the user to fill out page after page of questions, many of which may not be relevant or important to such user; (4) they take too much time to complete because they require cumbersome keyboard operations, and this results in poor data yield caused solely by user impatience; (5) they often require users to provide actual arithmetic or mathematical data input to represent data values perceived only subjectively by such users; in other words, if they ask a user to rate car characteristics, such person might have to assign a color parameter of a car as a "5," and a price parameter of such car as an "8". Later, the user might believe that the acceleration time is also important, and he/she would then be forced to compute some new value that is appropriate relative to the other numerical values already provided for other parameters. Furthermore, consideration of a new parameter might require scaling or re-adjustment of all prior values for other parameters. Such arithmetic manipulation is cumbersome and beyond the capability or interest level of many potential users of such programs.

It is apparent that many of these same problems are inherent in conventional objective data collection/presentation systems, to the extent they even utilize a graphical interface. Accordingly, such systems would benefit from a solution that ameliorates such problems.

SUMMARY OF THE INVENTION

The present invention, therefore, aims to provide a graphical interface which permits a user to select from a palette of preferences to describe those factors influencing his/her subjective opinion, actions, etc., relating to various items/events, and which allows such user to identify such factors to a data processing application efficiently, quickly, intuitively and without substantial human interaction;

An additional goal of the present invention is to provide a mechanism and method for application users to express subjective data in graphical, rather than arithmetic form, so that data entry procedures are simplified and made more efficient;

A further object of the present invention is to provide an apparatus and method for performing meaningful analyses of subjective data, by permitting application users to express subjective data in a graphical form that is nevertheless translatable into a data form that can be handled mathematically by an underlying application program;

Yet another object of the present invention is to provide a mechanism and method to permit a user to express a relative ranking of parameters pertaining to a particular topic/event in visual form, without requiring extensive data input, or human interaction/analysis;

Another object of the present invention is to provide a mechanism and method for users to express personal preference data items, as well as relationships between such items, through graphical tools which visually depict and/or prioritize such data items in a manner conducive to quick and easy understanding by the user;

A further object of the present invention is to provide a system and method for users to identify, store, recall and modify experiences, lessons and understandings gleaned from participating in various actions and transactions, so that they may learn and benefit from their past mistakes and successes when they adopt particular strategies for engaging such actions and transactions;

A preferred embodiment of an electronic interface of the present invention achieves these objects by permitting a user to communicate subjective data information concerning a proposed or actual action/transaction (i.e., such the user's mental

impressions of such event, an item of interest, or some lesson learned by such user associated with the same) to an underlying application program during what can be generally described as a "data painting sessions." The interface includes a parameter "menu," (or pallette) which menu provides a user with a visible set of data parameters which may be associated with the subjective data information. The parameters can be presented in text form, symbolic form, or some other form easily comprehendable by the user, and can be customized in part by the user to reflect individual preferences. In a separate portion of the interface a parameter "canvas," is presented so that it is simultaneously visible with the parameter menu. The user can select and manipulate the data parameters, placing them on the parameter canvas using a drag and drop motion, thus generating a kind of data "picture" for the action/transaction. This data picture can be stored, retrieved, edited and modified at will during a later session if desired. The interface is preferably configured so that all of the user's subjective data information is captured using only the parameter menu and canvas tools presented by such interface, and during a single data collection session.

Again in a preferred embodiment, the interface is configured so that the data parameters associated with the subjective data information are selected and moved by such user from the parameter menu to the parameter canvas using an an electronic pointing device. The menu and canvass are located proximately to each other so that the user can perform the act of moving the parameters to the canvass in rapid fashion.

At the end of the data painting session in this preferred embodiment, the identified data parameters associated with the subjective data information are stored as one or more electronic records corresponding to an electronic data picture. Notably, this data picture includes numeric data values, but is generated transparently without numeric data input by the user, thus reducing the burden experienced by the user in presenting his/her data selections. The numeric data values are based on the physical location of the data parameters as placed by the user on the parameter canvas, thus allowing the user to rank the parameters in relative importance quickly, easily, and without computational effort on their part. This relative ranking between data parameters can be changed by the user by

simply altering a relative physical arrangement of the data parameters on the parameter canvas using a computer mouse or similar pointing device. In certain applications for more skilled users, the data parameters can be ranked by both a relative horizontal and vertical location on said parameter canvas.

Further in the preferred embodiment, the parameter canvas includes a gradient visible to the user for assisting in the ranking of the data parameters. The gradient is presented in differing shades of color as reinforcing feedback. Additionally, while the user is arranging the data parameters, the parameter canvas conveys visible feedback information, alerting him/her to the action they are taking.

In another variation, and so as to save time and burden on the user, the interface can present an initial data picture to the user based on that person's prior data pictures. This initial data picture can then be modified as desired to reflect the particular action/transaction.

To further enhance the user's utilization of the interface, an additional chart or picture can be presented to them depicting various correlations between the data picture being created and prior data pictures made by such user. In this fashion, the user can determine at a glance what the outcome of a particular action/transaction is likely to be based on the variables identified at that time.

A preferred user data capture method of the present invention utilizes the aforementioned interface for permitting a user to identify personal parameters concerning an action and/or transaction to an underlying computer program.

Although the invention is described below in a preferred embodiment associated with a personal stock portfolio managing program, it will be apparent to those skilled in the art that the present invention would be beneficially used in many other applications where it is desirable to capture data in a fast, easy, and comprehensive manner that minimizes burden on a user providing such input.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow chart depicting the general operation of an embodiment of the present invention;

Figure 2 is a screen shot from a computer program application incorporating one embodiment of the interface of the present invention;

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 is a flow chart illustrating the basic operation of a data collection interface portion 100 of an application program incorporating the present invention. In a preferred embodiment the present interface is used by a stock portfolio managing program to elicit feedback and information concerning a user's motivations, opinions, reasonings, etc. for participating in a particular transaction - i.e., by either purchasing or selling a particular equity. This program is entitled "TotalTrader" and can be obtained from visiting a website maintained by the assignee at www.totaltrader.com.

As used herein, nonetheless, the terms "action" or "transaction" are intended in their broadest sense to mean an event of interest to the user that has already occurred or may occur in the future, or even an article or item of interest to the user. For instance, in the latter case, when the present interface is used in connection with an online marketing or sales research program, a transaction can represent a user's interest in a particular product or service being offered by a vendor, such as an automobile or travel agency services. Accordingly, at step 110, a transaction data picture is requested from the user of such program. This operation takes place, for example, after preliminary data concerning the transaction is first obtained, such as date of the purchase, number of shares, price per share and the like. After this, the user is presented at step 115a with a visual window identifying both a pallette of parameters (representing assertions, reasons, motivations, etc.) and at 115b with a data canvas for creating a customized data picture depicting the user's total rationale for the transaction in question. It should be noted that for ease of use, and as shown in FIG.2 , the palette and canvas are presented visually to the user at the same time. As will be apparent from the description below, the palette and canvas

present a simple, single data collection screen for the user to capture all of the information associated with a particular action/transaction during a particular session. The electronic windows for such palette and canvas are generated using well-known programming techniques, and the specifics of the same are not material to the present invention except as described herein. Nonetheless, a listing of the important source code routines used in the present invention is appended to the end of the present disclosure as Appendix A.

During an interactive session at step 120 (which can be thought of as a data "painting" session) the user is permitted to select from the list of parameters (which again can be reasons, assertions, etc.) presented, and to drag and drop any particular item as desired to a location on the data canvas. In this manner, the user can quickly, easily and intuitively manipulate subjective data parameters into a more concrete and structured format, representing a data "picture" without having to plow through numerous irrelevant data queries, and without having to maintain a precise mental arithmetic valuation (or relative ranking) of the data parameters selected. Instead, the vertical (and horizontal) placement of such data parameters as arranged within such window by the user can be used as ranking indicators. At the end of the data "painting" session at step 125 the data picture is converted into a series of data values in a fashion completely transparent to the user, and without any additional effort on their behalf. These data values are in turn either stored or passed on to an application program at step 130. The interface then yields control back to the main application program at 135.

FIG. 2 illustrates one embodiment of the inventive interface 200 as seen by a user at step 120. On the left hand side of the window, a menu list 205 of parameters 221, 222, etc. are identified. This set is presented visually to the user in what is known in conventional terms as a tree-structured menu. Preferably, for ease of use, this tree menu 205 has a shallow depth (i.e., number of levels) and reasonable breadth (i.e., number of items per level) that is manageable, and which permits a user to visualize all possible selections without having to perform time consuming scrolling operations. Generation of tree-structured menus is well-known in the art, and so will not be discussed at length

here. The use of a tree-structured menu 205 is especially advantageous within the context of the present invention, nonetheless, since it permits a user to rapidly identify those reasons affecting or motivating their behavior vis-a-vis the transaction in question. Again, in the present preferred embodiment, the reasons and assertions presented in menu 205 are those factors commonly associated with actions or transactions (buying or selling) securities or options. These include such factors as technical charting indicators, specific events that may affect the stock price (a split announcement for example), rumors, tips from friends, etc. These factors are broken into larger category groups at a first level 220, and in more fine detail at a sub-category second level 221. For instance, category group "Chart Pattern" 220 is broken down into 6 sub-categories 221 identified as "Inverted Head & Shoulders," "Cup with Handle," "Broke above Trendline," Broke below Trendline," "Broke below support," and "Rounded Top," all of which are commonly understood technical analysis tools in the field of financial instrument trading. It should be apparent that such category and sub-categories are merely illustrative of the parameters that could be identified in interface 200, and that the present invention is not limited to any specific set of the same. For example, in an interface geared towards understanding the motivations of a prospective consumer in the automotive field, category sets may include quality, warranty, price, performance, luxury, reliability, and so on. Sub-categories for price might be broken down into numerical sets ranging from 10k to 15k, 15k to 20k, 20-25k, 25-30k, etc. The particular implementation of the parameter set, of course, can be tailored to the features/functions associated with the item in question, so that reasonable allowance is made for variations in the characteristics of such item.

Another advantage of the present invention lies in the fact that the menu 205 can be supplemented by the user to include new categories 221 reflecting particular assertions, reasons, motivations, influences, etc., experienced by such user. As an example, a new category "other" 221 can be added to menu 205 to include such sub-categories as "Read in local newspaper," "Heard good things from John X," etc. Thus menu 205 can be customized in part, and this provides additional flexibility to accommodate particular influences in the user's realm of experience. Such additions to

menu 205 can be effectuated using any number of known techniques in the art.

Accordingly, interface 200 presents the user with a complete and customized set 205 of parameters that are applicable to the transactions handled by an underlying application program. Moreover, such parameters are depicted in a manner that avoids prior art cumbersome multiple-page formats, which tend to confuse and slow down interaction between the user and the application program. The arrangement of menu 205 in fact, presents the user essentially with an electronic data pallette in a first portion of the interface, which he/she can easily visualize in complete form at a glance and utilize to create a data picture 210 on a data canvas 215 (or preference field) in a second portion of the interface as shown on the right hand side of interface 200.

In the example given in FIG.2, data picture 210 includes two subjective parameters 230, 231 selected by a user from menu 205 as motivating his/her to purchase a particular stock. In this instance, such user has identified certain assertions as the basis for his/her action, including the fact that they are influenced by their perception that the company has a "Good Dividend," and that the "Company has good products." The selection of parameters 230 and placement in preference field/data canvas 215 is done with a conventional computer mouse in a drag and drop manner well-known in the art, and as illustrated generally in FIG.2 by the path of the dotted lines shown there. Other techniques, such as direct control devices (light pens, touch-screens, and so on) or indirect control devices (touch pads, joysticks, trackballs etc.), can also be used in lieu of a mouse for the painting of data picture 210. The paramount consideration, in this respect, is that an easily manipulable tool should be selected to permit the user to interact quickly and intuitively with palette 220 and move objects (data parameters) to data canvas 215 with ease.

In this preferred embodiment, preference field/data canvas 215 is presented in a relevance spectrum format, with descriptors at the top/bottom providing visual feedback to the user to identify a value of importance for the identified parameter, ranging from "Less" to "More" important. This allows the user to not only identify the data parameters associated by them in connection with the transaction, but also to rank such parameter in

a more objective fashion, both absolutely in terms of a perceived importance, but also relative to other factors. This ranking, too, can be done quickly by the user since it is done without the need for cumbersome arithmetic computations and/or manipulation of electronic sliding scales, etc. In fact, the only physical operations required of the user are the selection (preferably using a common computer mouse) of parameters on the left hand side of the interface, and their drag and drop placement on the right hand side of the interface. As the pallette 205 and canvas 215 are located closely and adjacent to each other spatially, this task can be performed extremely ergonomically and rapidly by a user utilizing a computer mouse. In most cases, in fact, the distance required for the user to move a data parameter from palette 205 to canvas 215 is relatively small: i.e., less than half the total interface screen size. This reduction in demands on the user's attention and time results in more thorough and accurate data extraction because the user is not discouraged or de-motivated by the interface from presenting his/her opinions in complete form, and in a manner that is more intuitive and in tune with the way in which most human operators work most comfortably - namely, with a "picture" of their impressions, rather than in hard numerical representations. Moreover, after such data is captured, it can be retrieved and modified at will by the user, permitting them to perfect and learn from their mental impressions of actual or proposed actions/transactions.

These advantages of the present invention can be illustrated very easily with simple example. In a typical prior art program interface, the user is required to parse through page after page of questions, assigning numerical values to each parameter of interest. This is slow because the user is confronted with an ocean of choices that may be of no importance, but which yet must still be navigated to satisfy the data collection goals of the underlying application. In the present invention, the user need only utilize those data parameters of interest to them from a data parameter pallette. Then, as the user traverses a typical prior art interface, they may assign a number of relevance values R1, R2...Rk, and the like for each identified parameter. A substantial mental burden is imposed on the user in both remembering the relevance values previously provided for prior parameters, and in attempting to effect a relative ranking of new parameters within

the framework of the data collection effort at that point. In other words, the user may associate two parameters with the same relevance, but if information on the two parameters is solicited at different times during the data collection interview, he/she is likely to forget the value previously assigned. This means that the data value given may be inaccurate, and the data collection effort is degraded. Furthermore, when confronted with the fact that he/she has previously identified two parameters as having a relative ranking of say 6 and 9, a third parameter lying between these two in importance must be assigned an arithmetic value that is calculated mentally by the user. A value for a fourth parameter, perhaps associated by the user between the third and second parameter in importance, must then be calculated by the user. It is apparent that this mental effort is both time consuming and potentially frustrating. In contrast, in the present invention, users can easily and dynamically arrange and re-arrange their identified priorities through simple physical manipulation, and without the need for tagging these parameters with hard, fixed arithmetic values. Instead, the underlying application program can simply process the data picture provided, and then carry out a computational evaluation of the necessary values to be associated with the data items arranged in the data picture. This fact, too, reduces significantly the time associated with a data interview, and is likely to result in greater user appreciation for (and utilization of) the underlying application programs.

A further advantage provided by the present invention lies in the fact that since no keyboard typing entries are required of the user, and no electronic "check boxes" or the like are filled in, errors in the data capture process are also reduced, thus further increasing the reliability of data captured using the inventive interface. In fact, the possibility of errors is minimized in large part because the user is permitted to see the entirety of his/her data entry visually for sufficient periods of time to allow for detection of obvious errors. This and other aspects of the present invention therefore make optimal use of visual information processing centers available to human operators.

Finally, a useful industry accepted predictive model of hand movement complexity in an interface (i.e., moving a pointing device to one region of a display to

another) is provided by a formula in Schneiderman, at page 325. This formula states that:

Index of difficulty = $\text{Log}_2(2D/W)$.

In this formula, D = distance that must be traveled across the screen by a pointing device (such as a mouse), and W is the area of the target for the pointing device. From this simple model, it is readily apparent that the interface proposed by the applicant is optimized to reduce user interaction difficulties. This is because, as seen in FIG.2, the travel distance (D) for the user's mouse in moving data parameters from the menu to data canvas 215 is kept very small; conversely, the area (W) for data canvas 215 is relatively large. All things being equal, this mathematical model illustrates why an index of difficulty for the present invention is relatively small as compared to prior art interfaces which rely on the operator's ability to move a cursor across an entire screen (to address an item of interest) and then require fairly accurate control to land within a small designated area for expressing some value for such item of interest. A time for an operator to achieve such movement, of course, is directly related to such index as indicated by Schneiderman; again, for the present invention, this predictive modeling helps explain why an interface of the type described herein is much faster for even a novice operator, and thus more likely to be endorsed by the same.

Similarly, at pp. 391 - 397, Schneiderman addresses the issue of display layout "appropriateness," as measured by a series of metrics discussed therein, including task frequency and sequencing characteristics. An evaluation of the layout of the present invention by such metrics reveals, quite distinctly, that an optimal arrangement is provided from the user's perspective, because task sequences are kept extremely simple (the user need only perform one physical operation to input and classify a data parameter), and the physical arrangement of the interface (relative placement of data menu and data canvas) is implemented such that such physical operation is effectuated by a single motion that occurs over a very short distance, and with high accuracy. This minimizes the amount of visual scanning (and thus time and burden) required by the user to express his/her thoughts, because an interaction area is minimized as well. Since related ideas are grouped in physically contiguous areas in menu 205, this further reduces

eye strain, mental strain, etc.

To further enhance the appearance and utility of data canvas 215, color, shading, highlighting and other visual indicators can be used to provide feedback to the user as well. In a preferred embodiment data canvas 215 includes a blue background, and is arranged so that a "gradient" is presented to the user in the form of shading arranged from darkest (more important) to lightest (least important). This arrangement is chosen to provide the user with a pleasant visual environment, and as a visible metaphor/reinforcer for the fact that factors weighing "heavier" in his/her mind should be placed in a heavier shade portion of the gradient. In addition, as individual data parameters 230 are moved on data canvas 215, nearby already-placed data parameters (or the appropriate spectrum legend at one end of the gradient), can "glow" or become highlighted to remind the user that they are moving the selected data parameter to a region of relevance that is close to the previously identified data parameter. This feature, too, helps the user to orient and rank his/her reasons, preferences, opinions, etc. in a more orderly and reliable fashion, and without the need for arithmetic computations. Another visual enhancement that could be made, of course, is the addition of scaling information - for instance, some form of ruler markings as used in conventional word processing layout programs, or drafting programs - along the edges of canvas 215. Such markings could include numerical indicators ranging, for example, from 1 to 10, or other convenient divisions to better identify the gradient. Other variations are of course possible, and will apparent to those skilled in the art based on studies of human perceptual skills and traits and from the teachings herein.

When the user has completed the creation of data picture 210, it can then be saved and stored using conventional techniques as a transaction "reasons" file (or entry) for later use by an underlying application program. A conversion takes place (see FIG.1, step 125) so that the user data picture can be utilized by an underlying application program. The specifics of such conversion will of course vary from application to application, depending on the nature of the specific data parameters presented to the user, the nature of the specific data canvas, etc, etc. In the present embodiment, an identification is first

made of all the data parameters 215 making up data picture 210 on data canvas 215. The placement of such data parameter within data canvas 210 is also noted; in this instance, only the vertical distance component is retained, but it should be apparent that a horizontal component could also be used.

Thus, a typical data picture record can include a transaction identifier, the identity of a data parameter, and a location placement within the gradient (in arbitrary units). Other information can of course be included in such record and many different variations are possible for such data picture record from application to application. Each data picture 210, therefore, will consist of one or more of such data picture records, thus constituting a complete transaction record reflecting a complete capture of the user's perceptions, motivations, reasoning, etc., for a particular transaction. At this point, as indicated at step 130 (FIG. 1) transaction record, and all of its underlying components are available for use by an applications program as needed. In a preferred embodiment, such transaction records are maintained and retrievable at later times by an operator so that personalized lessons can be learned from correlations of the user's data pictures (describing the user's particular rationale for a transaction) and resulting gain or loss from such particular stock purchase or sale transactions.

It should be noted that in addition to the vertical placement component value retained, a horizontal placement component might be used, for example, where data canvas 210 also has a horizontal spectrum (left to right orientation) depicting a different consideration, i.e., such as the user's perception of the credibility of such identified parameter. As an example, a user might identify a rumor as strongly motivating his/her behavior (thus placing the data parameter high on the data canvas), but could also qualify this belief by indicating that the perceived truthfulness of the rumor is low by placing the data parameter on the far left (or right) according to an "accuracy" spectrum. This feature permits the present interface to better capture the "fuzzy" logic of human actions and perceptions through an additional variable qualifying the data parameters identified for a particular transaction.

For purposes of the present invention a detailed description of those features

commonly found and understood in application windowing technology (i.e., such as sizing, scrolling, handling and the like) is not provided. Such features can be implemented in any one of many techniques known in the art, and the invention is not limited in any way by such specific implementations.

Consequently, the present invention permits a user of an application program to enter data parameters in an efficient, intuitive, rapid fashion using a single data collection window which captures all of the subjective information in a single snap shot. And, through the manipulation of only a few visual objects representing individual perceptions, motivations, reasons, etc., an underlying application program can capture more relevant data more efficiently than through the use of prior art interfaces.

In addition, the present invention allows, for the first time, for a user to convey his/her mental impressions in a form understood by him/her, and yet parseable in an intelligent fashion by an underlying program. The present inventive interface, in fact, can serve as a simple, non-intimidating environment and mechanism to permit even novice users to interact with sophisticated and complex application programs that would otherwise be avoided. This interaction is expected to increase as application programs begin to make more and more use of "fuzzy" data passed on by users of the same.

In another variation of the present invention, the underlying application program can track prior transaction reasons records created by the user. From this tracking, an evaluation can be made of the N most common data parameters identified by the user, as well as their average placement on data canvas 215. This information, in turn can be used to generate an "expected" data picture 210,' which, when a new transaction reasons record is to be generated (i.e., at step 110) can be presented to the user (at his/her option) as a template to work from. At that point, the user can accept, modify, supplement such data picture 210' based on the particular details of the transaction in question. In this manner, the burden on a user to input relevant data is further minimized.

For added functionality, the underlying application program can also dynamically process data picture 210 into a transaction record, and then visually display a correlation of such transaction record with other transaction records 210' previously entered by the

user. In this manner a user can quickly and visually identify similarities, trends, etc. in his/her rationale (or other people's rationales) for such transactions. This feedback window 260 can be generated using any one of conventional programming techniques based on the nature of the underlying data, the perspective desired by the user, the desired sub-set of prior transaction records, the desired correlation, etc., and can be presented alongside graphical interface 200 as illustrated generally in FIG. 2. As a simple example, in the case of a financial trading context, the user could request a comparison chart in feedback window 260 illustrating the overall financial gain/loss in a graph 261 incurred by such user (based on evaluation of prior transaction records) when they (or other users) had identified "Good Dividend," and the "Company has good products" (or some other reasons) as reasons for their purchasing a particular financial instrument. The overall average could be plotted as well as a separate graph 262 as noted. Other variations of feedback window 260 and information to be displayed thereon are of course possible, and the present invention is not limited to any particular implementation. For example, instead of a chart, a simple icon resembling a human face smiling or frowning can communicate positive or negative feedback information, letting the user know immediately whether they are getting closer or further away from an ideal data picture. This aspect of the present invention allows the user to immediately visualize the expected results of the action/transaction based on that person's unique experiences/rationales. This feature is extremely beneficial as it can serve to prevent obvious errors, or to suggest a course of action that is likely to be favorable based on an evaluation of prior transaction data. Other potential applications which can utilize this kind of immediate visual feedback including telemarketing, product service support, etc. In such environments, the present interface could be used as a marketing data capture interface, and as data is collected by an operator from a customer, feedback window 260 can present potential options to the operator correlating such user's data with prior data transactions, or against some set of criteria. As one example, in response to various customer descriptions of problems with a product, an operator could be given a visual or symbolic list of potential areas that are the origin of the same. This is but one example, of course, and the present

invention can be advantageously used in any number of similar environments.

In yet another variation, data parameters 220 can of course be presented in symbolic, or iconic form, rather than as descriptive text objects. For example, in an application program soliciting feedback on travel modes used by a prospective traveler, menu 220 may instead contain a series of graphical icons representing conventionally accepted depictions of trains, planes, cars, taxis, bicycles, cruise ships, etc. In an application program for capturing entertainment interests, iconic representations of movies, Cds, LP records, television, etc. could be presented as data parameters. Other examples will be apparent to those skilled in the art for other fields of interest.

In another embodiment of the inventive interface 200, instead of corresponding to "reasons" employed by a user for a particular action, parameters 221, 222, etc. in menu 205 correspond generally to "lessons" learned from a particular action/transaction. This way, during an interactive session at step 120 the user is permitted to paint a data picture explaining what he/she learned from the particular experience of engaging in the action/transaction. Preferably this second variation of the inventive interface is used in conjunction with, and as a complement to the first embodiment noted above. For example, after first creating a data picture with the aforementioned graphical interface to identify particular reasons and motivations for a particular action/transaction, the user can then invoke a separate interface at a later time for identifying any subsequent lessons, understandings, etc. that he/she has acquired or associates with the particular action/transaction. This session then results in a second data picture 210 associated with lessons specific to such user which they can store, modify, and learn from to improve their objectivity when participating in such actions/transactions. As an example, in the case of a financial trading context, parameters 205 can list such options as "sold too early," "sold too late," "got in too early," "got in too late," "don't listen to advice from X," etc. This information would be captured by the interface so the user can maintain a permanent but modifiable diary or log of useful data which can be referred to at a later time to evaluate their performance, to see what weaknesses or strengths they exhibit, to see what events seem to influence their thinking, and to observe trends in the same.

As is apparent, for ease of use for the user, this second interface only varies from the first interface described above in the identity of the parameters provided in menu 205, which, again, need only identify particular "lessons" instead of "reasons" in this case. If desired, data canvas 215 can also be modified nonetheless with suitable language adjustments to the spectrum legend to correlate better with the parameters provided in menu 205. Again, as with the first interface, the user can rank the relative perceived importance of the lessons learned, which acts as strong visual feedback when reviewed at a later time.

To better capture the manner in which individuals collect information, the data pictures 210 created by the second interface can also be modified at a later time to reflect new insights garnered by the user. For instance, after closing out a losing transaction involving an equity purchase and sale, the user may identify the fact that they bought too early as a lesson that they learned. At a later time, in the event the equity price rebounds, they may also conclude that they sold too early as well. Because the data picture is stored electronically, it can be retrieved via interface 200 and modified at will by the user to reflect such new understandings and lessons about the action/transaction. This functionality permits the user to better grasp his/her strengths and weaknesses at a glance, and helps reinforce the lessons learned from such actions/transactions.

While the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. For example, it is apparent that the present invention would be beneficial used in any applications program where it is desirable to obtain accurate, reliable information from a user in an efficient, intuitive fashion. Other types of particular implementations for the data parameter menu and data canvas beyond those illustrated in the foregoing detailed description can be used suitably with the present invention. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

APPENDIX A

```
Content-Type: text/java;
        name="ReasonDragPanel.java"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
        filename="ReasonDragPanel.java"
import java.awt.*;
import java.util.Vector;
public class ReasonDragPanel        extends Panel
{
        protected Image im1;   // offscreen image
        protected Graphics g1 =3D null;  // offscreen graphics context
        private int viewHeight =3D 300;
        private int viewWidth  =3D 300; // pixel size of tree display
        private Color bgHighlightColor =3D Color.blue;   // selection bg color
        private Color fgHighlightColor =3D Color.white;   // selection fg color
        private Color fgColor =3D Color.black;
        private Color bgColor =3D Color.white;
        private Vector drags;
        private FontMetrics fm; // current font metrics
        private int mouseX =3D -1;
        private int mouseY =3D -1;
        private int grabX =3D 0;
        private int grabY =3D 0;
        private DragTreeNode selectedNode =3D null;
        private boolean dragging =3D false;
        private boolean dragUp =3D false;
        private boolean dragDown =3D false;
        private Color topColor =3D new Color(128, 128, 255);
```

```
        private Color bottomColor =3D new Color(255, 255, 255);
        private int fadeSteps =3D 20;
        public String topLabel =3D "More Important";
        public String bottomLabel =3D "Less Important";
=09
        public ReasonDragPanel()
        {
                drags =3D new Vector(5, 1);
        }
        public Vector getDrags()
        {
                return drags;
        }
        public void addNode(Reason r, int y)
        {
                for (int i =3D 0; i < drags.size(); i++)
                        if
(((DragTreeNode)drags.elementAt(i)).reason.name.equals(r.name))
                                return;
                for (int i =3D 0; i < drags.size(); i++)
                        if (!((DragTreeNode)drags.elementAt(i)).dragged)
                                drags.removeElementAt(i--);
                drags.addElement(new DragTreeNode(r, 10, y, !(y =3D=3D 10)));
                repaint();
        }
        Color fadeColors(Color f, Color b, double per)
        {
                double nper =3D 1 - per;
                int r1 =3D (int)(f.getRed() * per + b.getRed() * nper);
```

```
                int g1 =3D (int)(f.getGreen() * per + b.getGreen() * nper);
                int b1 =3D (int)(f.getBlue() * per + b.getBlue() * nper);
                return new Color(r1, g1, b1);
        }
        public void update (Graphics g)
        {
                redraw();
          paint(g);
        }
=09
        public void paint (Graphics g)
        {
                Dimension d =3D size();
=09
                if ((d.width !=3D viewWidth) || (d.height !=3D viewHeight))
                        redraw();
                g.drawImage(im1, 0, 0, this);
        }
        public void redraw()
        {
                drawDrag();
        }

        public void drawDrag()
        {
                Dimension d =3D size();
=09
                if ((d.width !=3D viewWidth) || (d.height !=3D viewHeight) || =
(g1=3D=3Dnull))
```

```
            {
                im1 =3D createImage(d.width, d.height);
                if (g1 !=3D null) {
                    g1.dispose();
                }
        g1 =3D im1.getGraphics();
        viewWidth=3Dd.width;
        viewHeight=3Dd.height;
          }
=09
    Font f =3D null;//getFont();  //unix version might not provide a = default font
    if (f =3D=3D null)
    {
      f =3D new Font("Helvetica", Font.PLAIN, 13);
      g1.setFont(f);
      setFont(f);
    }
        if (f !=3D null)
        {
            if (g1.getFont() =3D=3D null)
                g1.setFont(f);
        }
  =09
      fm =3D g1.getFontMetrics();
      g1.setColor(bgColor);
      g1.fillRect(0,0,viewWidth,viewHeight);  // clear image
        /*for (int i =3D 0; i < fadeSteps; i++)
        {
            g1.setColor(fadeColors(bottomColor, topColor, 1.0* i/fadeSteps));
```

```
                    g1.fillRect(0, size().height * i/fadeSteps, size().width, =
1+size().height / fadeSteps);
                }*/
=09
=09
                if (!dragging)
                    selectedNode =3D null;
                for (int i =3D 0; i < drags.size(); i++)
                    drawNode((DragTreeNode)drags.elementAt(i));
                /*g1.setColor(Color.red);
                for (int i =3D 0; i <=3D 100; i+=3D10)
                    g1.drawString(""+i, 20, valueToY(i));*/
                //if (dragging)
                {
               f =3D new Font("Helvetica", Font.BOLD + Font.PLAIN, 14);  g1.setFont(f);
                    if (dragUp)
                        g1.setColor(Color.red);
                    else
                        g1.setColor(fgColor);
                    g1.drawString(topLabel, (size().width - fm.stringWidth(topLabel))
>> =
1, fm.getHeight() + 2);
                    if (dragDown)
                        g1.setColor(Color.red);
                    else
                        g1.setColor(fgColor);
                    g1.drawString(bottomLabel, (size().width - =
fm.stringWidth(bottomLabel)) >> 1, size().height - fm.getHeight() + 2);
                }
```

```
            //g1.setColor(fgColor);
            //g1.drawRect(0,0,viewWidth, viewHeight - 1);
        }
        private int topLine()
        {
            Font f2 =3D g1.getFont();
          Font f =3D new Font("Helvetica", Font.BOLD + Font.PLAIN, 14);
          g1.setFont(f);
            int retVal =3D fm.getHeight() + 5;
            g1.setFont(f2);
            return retVal;
        }
        private int dragHeight()
        {
            return size().height - (3*topLine() + 4);
        }
        private int valueToY(int val)
        {
            return topLine() + 4 + (dragHeight() * val / 100);
        }
=09
        private void drawNode(DragTreeNode node)
        {
            int width =3D fm.stringWidth(node.reason.name);
            int height =3D fm.getHeight() + 4;
            boolean mouseIn =3D false;
            int realy =3D valueToY(node.y);
            int realx =3D node.x;
        =09
```

```
if (dragging && (selectedNode =3D=3D node))
{
        realx -=3D grabX;
        realy -=3D grabY;
}
g1.setColor(fgColor);
if (((mouseX >=3D realx) && (mouseY >=3D realy) &&=20
        (mouseX < realx + width+10) && (mouseY < realy + height+4))
||=20
        (dragging && (selectedNode =3D=3D node)))
{
        mouseIn =3D true;
        if (!dragging)
        {
                selectedNode =3D node;
                grabX =3D mouseX - realx;
                grabY =3D mouseY - realy;
        }
}
if (mouseIn)
        g1.setColor(bgHighlightColor);
else
        g1.setColor(bgColor);
g1.fillRect(realx, realy, width + 10, fm.getHeight() + 4);
if (mouseIn)
        g1.setColor(fgHighlightColor);
else
        g1.setColor(fgColor);
g1.drawRect(realx, realy, width + 10, fm.getHeight() + 4);
```

```
                g1.drawString(node.reason.name, realx+5, realy+fm.getHeight());
        /*      if (dragging && (selectedNode =3D=3D node))
                {
                        g1.setColor(Color.red);
                        g1.drawString(""+node.y, realx, realy+2+(fm.getHeight() * 2));
                }*/
        }
=09
=09
=09
=09

        public synchronized Dimension preferredSize()
        {
          return new Dimension(175, 125);
        }
=09
        public synchronized Dimension minimumSize()
        {
          return new Dimension(50, 50);
        }
        public boolean mouseMove(Event e, int x, int y)
        {
                mouseX =3D x;
                mouseY =3D y;
                repaint();
                return true;
        }
        public boolean mouseDrag(Event e, int x, int y)
        {
                dragUp =3D (y < mouseY);
```

```
                dragDown =3D (y > mouseY);
                if (selectedNode !=3D null)
                {
                        dragging =3D true;
                        selectedNode.dragged =3D true;
                        selectedNode.x =3D x;
                        selectedNode.y =3D 100 * (y-topLine()) / dragHeight();
                }
                repaint();
                return true;
        }
        public boolean mouseUp(Event event, int x, int y)
        {
                if (dragging)
                {
                        selectedNode.x -=3D grabX;
                        selectedNode.y -=3D (100 * grabY / dragHeight());
                        if (selectedNode.y < 0)
                                selectedNode.y =3D 0;
                        if (selectedNode.y > 100)
                                selectedNode.y =3D 94;
                        if ((selectedNode.x < 0) && (selectedNode.x + =
fm.stringWidth(selectedNode.reason.name) + 10) > 0)
                                selectedNode.x =3D 0;
                        if ((selectedNode.x < size().width) && (selectedNode.x + =
fm.stringWidth(selectedNode.reason.name) + 10 > size().width))
                                selectedNode.x =3D size().width - =
(fm.stringWidth(selectedNode.reason.name) + 10);
                        Rectangle r =3D new Rectangle(size());
```

```
                    Rectangle r2 =3D new Rectangle(selectedNode.x, selectedNode.y,
                                fm.stringWidth(selectedNode.reason.name) + 10,
fm.getHeight()+4);
                    if (!r.intersects(r2))
                        deleteNode(selectedNode.reason.id);
                    selectedNode =3D null;
                    dragging =3D false;
                    dragUp =3D false;
                    dragDown =3D false;
                    repaint();
                }
            return true;
        }
        public void deleteNode(int id)
        {
            for (int i =3D 0; i < drags.size(); i++)
            {
                if (((DragTreeNode)drags.elementAt(i)).reason.id =3D=3D id)
                    drags.removeElementAt(i);
            }
        }
}
Content-Type: text/java;
        name="ReasonTree.java"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
        filename="ReasonTree.java"
import java.awt.*;
import java.util.Vector;
```

```
import TreeNode;
public class ReasonTree       extends Panel
{
        public static final int CHILD  =3D 0;
        public static final int NEXT   =3D CHILD + 1;
        public static final int LAST   =3D CHILD + 2;
        public static final int SEL_CHANGED =3D 1006; //selection changed event
=09
        private TreeNode rootNode;        // root node of tree
        private TreeNode selectedNode;    // highlighted node
        private TreeNode topVisibleNode;  // first node in window
=09
        Scrollbar  sbV;                        // vertical scrollbar
        int                    sbVPosition=3D0;    // hold value of vertical scrollbar
        int                    sbVWidth;     // width of vertical scrollbar
        long           sbVTimer =3D -1;     // time of last vert scrollbar event
        private int count=3D0;   // Number of nodes in the tree
        private int viewCount=3D0;// Number of viewable nodes in the tree
        private Color bgHighlightColor =3D Color.blue;   // selection bg color
        private Color fgHighlightColor =3D Color.white;  // selection fg color
        private int viewHeight =3D 300;
        private int viewWidth  =3D 300; // pixel size of tree display
        private int viewWidest =3D 0 ;  // widest item displayable (for horz =
scroll)
        int cellSize   =3D 16;  // size of node image
        int clickSize  =3D 8;   // size of mouse toggle (plus or minus)
        int imageInset =3D 3;   // left margin of node image
        int textInset  =3D 6;   // left margin for text
        int textBaseLine=3D 3;  // position of font baseline from bottom of =
```

```
cell
        private FontMetrics fm; // current font metrics
        long timeMouseDown;     // save time of last mouse down (for double =
click)
        int doubleClickResolution =3D 333; // double-click speed in =
milliseconds
        Portfolio applet;
        protected Image im1;    // offscreen image
        protected Graphics g1 =3D null;  // offscreen graphics context
        int prevX =3D -1;
        int prevY =3D -1;
        private String descrip =3D null;
=09
        public ReasonTree(Portfolio a)
        {
           super.setLayout(new BorderLayout());
           add("East",  sbV =3D new Scrollbar(Scrollbar.VERTICAL));
                applet =3D a;
        }
        public void clearTree()
        {
                rootNode =3D null;
                selectedNode =3D null;
                topVisibleNode =3D null;
        }
        public void setBackground(Color c)
        {
           super.setBackground(c);
           repaint();
```

```
    }

    public Color getBackground()
    {
      return super.getBackground();
    }
=09
=09
    public void setForeground(Color c)
    {
      super.setForeground(c);
      repaint();
    }
=09
=09
    public Color getForeground()
    {
      return super.getForeground();
    }
=09
=09
    public void setFgHilite(Color c)
    {
      fgHighlightColor =3D c;
      repaint();
    }
=09
    /**
     * Gets the current foreground selection hilite color.
```

```
     * @return the current foreground selection hilite color
     * @see #setFgHilite
     */
    public Color getFgHilite()
    {
      return fgHighlightColor;
    }
=09
    /**
     * Gets the current background selection hilite color.
     * @return the current background selection hilite color
     * @see #getBgHilite
     */
    public void setBgHilite(Color c)
    {
      bgHighlightColor =3D c;
      repaint();
    }
=09
    /**
     * Gets the current background selection hilite color.
     * @return the current background selection hilite color
     * @see #setBgHilite
     */
    public Color getBgHilite()
    {
      return bgHighlightColor;
    }
=09
```

```
        public void insert(TreeNode newNode, TreeNode relativeNode, int =
position)
        {
          if (newNode=3D=3Dnull || relativeNode=3D=3Dnull)
            return;
          =20
          if (exists(relativeNode)=3D=3Dfalse)
            return;
          =20
          switch (position)
          {
            case CHILD:
              addChild(newNode, relativeNode);
              break;
              =09
            case NEXT:
              addSibling(newNode, relativeNode, false);
              break;
              =09
            case LAST:
              addSibling(newNode, relativeNode, true);
              break;
=09
            default:
              // invalid position
              return;
          }
        }
=09
```

```
    /**
     * Returns the "root" node. The root node is the top-most node
     * in the tree hierarchy. All other nodes stem from that one.
     * @return the root tree node
     */
        public TreeNode getRootNode()
        {
           return rootNode;
        }
=09
        /**
         * Returns the total number of nodes in the tree.
         */
        public int getCount()
        {
           return count;
        }
=09
        /**
         * Returns the total number of viewable nodes in the tree.
     * A node is viewable if all of its parents are expanded.
     */
        public int getViewCount()
        {
           return viewCount;
        }
=09
        /**
         * Determines if the given node is viewable.
```

```
     * A node is viewable if all of its parents are expanded.
     * @param node the node to check
     * @return true if the node is visible, false if it is not
     * @see #viewable(java.lang.String)
          */
         boolean viewable(TreeNode node)
         {
           for (int i=3D0; i<viewCount; i++)
           {
             if (node =3D=3D v.elementAt(i))
             {
               return true;
             }
           }
=09
           return false;
         }
=09
         /**
          * Determines if the node with the given text is viewable.
     * A node is viewable if all of its parents are expanded.
     * @param s the node text to find
     * @return true if the node is visible, false if it is not
     * @see #viewable(TreeNode)
          */
         boolean viewable(String s)
         {
           if (s=3D=3Dnull)
           {
```

```
            return false;
          }
=09
          for (int i=3D0; i<viewCount; i++)
          {
            TreeNode tn =3D (TreeNode)v.elementAt(i);
=09
            if (tn.getText() !=3D null)
            {
              if (s.equals(tn.getText()))
              {
                return true;
              }
            }
          }
=09
          return false;
        }
=09
        /**
         * Determines if the given node is in the ReasonTree.
     * @param node the node to check
     * @return true if the node is in the ReasonTree, false if it is not
     * @see #exists(java.lang.String)
         */
        public boolean exists(TreeNode node)
        {
          recount();
=09
```

```
            for (int i=3D0; i<count; i++)
            {
               if (node =3D=3D e.elementAt(i))
               {
                  return true;
               }
            }
=09
            return false;
         }
=09
         /**
          * Determines if the node with the given text is in the ReasonTree.
          * @param s the node text to find
    * @return true if the node is in the ReasonTree, false if it is not
    * @see #exists(TreeNode)
          */
         public boolean exists(String s)
         {
            recount();
=09
            if (s=3D=3Dnull)
            {
               return false;
            }
=09
            for (int i=3D0; i<count; i++)
            {
               TreeNode tn =3D (TreeNode)e.elementAt(i);
```

```
=09
            if (tn.getText() !=3D null)
            {
               if (s.equals(tn.getText()))
               {
                  return true;
               }
            }
=09
         return false;
      }
=09
      // add new node to level 0
      /**
       * Adds a new node at root level. If there is no root node, the given
       * node is made the root node. If there is a root node, the given node
       * is made a sibling of the root node.
       * @param newNode the new node to add
       * @see #insert
       */
      public void append(TreeNode newNode)
      {
         if (rootNode=3D=3Dnull)
         {
            rootNode=3DnewNode;
            selectedNode =3D rootNode;
            count=3D1;
         }
```

```
            else
            {
               addSibling(newNode, rootNode, true);
            }
         }
=09
         void addChild(TreeNode newNode, TreeNode relativeNode)
         {
            if (relativeNode.child =3D=3D null)
            {
               relativeNode.child =3D newNode;
               newNode.parent =3D relativeNode;
               count++;
            }
            else
            {
               addSibling(newNode, relativeNode.child, true);
            }
=09
            relativeNode.numberOfChildren++;
         }
=09
         void addSibling(TreeNode newNode, TreeNode siblingNode)
         {
               addSibling(newNode,siblingNode,true);
         }
=09
         void addSibling(TreeNode newNode, TreeNode siblingNode, boolean =
asLastSibling)
```

```
        {
            if (asLastSibling)
            {
                //Find last sibling
                TreeNode tempNode =3D siblingNode;
                while (tempNode.sibling !=3D null)
                    tempNode =3D tempNode.sibling;
            =09
                tempNode.sibling =3D newNode;
            }
            else
            {
                //Insert the newNode below the siblingNode
                newNode.sibling =3D siblingNode.sibling;
            =09
                siblingNode.sibling =3D newNode;
            }
        =09
            //Set the parent of the new node to the parent of the sibling
            newNode.parent =3D siblingNode.parent;
        =09
            count++;
        }
    =09
    =09
        private Vector e; // e is vector of existing nodes
        private void recount()
        {
          count =3D 0;
```

```
        e =3D new Vector();
=09
        if (rootNode !=3D null)
        {
5           rootNode.depth=3D0;
            traverse(rootNode);
        }
    }
=09
10  private void traverse(TreeNode node)
    {
        count++;
        e.addElement(node);
=09
15      if (node.child !=3D null)
        {
            node.child.depth =3D node.depth+1;
            traverse(node.child);
        }
20      if (node.sibling !=3D null)
        {
            node.sibling.depth =3D node.depth;
            traverse(node.sibling);
        }
25  }
=09
    private Vector v; // v is vector of viewable nodes
    private void resetVector()
    {
```

```
                // Traverses tree to put nodes into vector v
                // for internal processing. Depths of nodes are set,
                // and viewCount and viewWidest is set.
                v =3D new Vector(count);
                viewWidest=3D30;
        =09
                if (count < 1)
                {
                        viewCount =3D 0;
                        return;
                }
        =09
                rootNode.depth=3D0;
                vectorize(rootNode,true,v);
                viewCount =3D v.size();
        }
=09
        private void vectorize
                        (TreeNode       node,
                         boolean        respectExpanded,
                         Vector                 nodeVector)
        {
          if (node =3D=3D null)
            return;
        =09
          nodeVector.addElement(node);
        =09
          if ((!respectExpanded && node.child !=3D null) || =
node.isExpanded())
```

```
        {
          node.child.depth =3D node.depth + 1;
          vectorize(node.child,respectExpanded,nodeVector);
        }
=09
        if (node.sibling !=3D null)
        {
          node.sibling.depth =3D node.depth;
          vectorize(node.sibling,respectExpanded,nodeVector);
        }
      }
=09
=09
      public boolean handleEvent(Event event)
      {
        if (event.target =3D=3D sbV)
        {
                if (descrip !=3D null)
                {
                        descrip =3D null;
                        redraw();
                }
          if (event.arg =3D=3D null)
          {
            return false;
          }
=09
          if (sbVPosition !=3D ((Integer) event.arg).intValue())
          {
```

```
                sbVPosition =3D ((Integer) event.arg).intValue();
                redraw();
            }
        }
            //if ((event.target =3D=3D this) && (event.id =3D=3D =
Event.MOUSE_DOWN))
            //      return mouseDown(event, event.x, event.y);
=09
        return(super.handleEvent(event));
            //return false;
    }
=09

  public void dragNode(int currX, int currY)
  {
    Graphics g =3D getGraphics();

    g.setXORMode(getBackground());
    if (prevX > 0)
            {
      g.drawRect( prevX, prevY, =
g.getFontMetrics().stringWidth(selectedNode.reason.name) + 4, =
g.getFontMetrics().getHeight() + 4);
                //g.drawString(selectedNode.reason.name, prevX+2, =
g.getFontMetrics().getHeight() + 2);
            }
    prevX =3D currX;
            prevY =3D currY;
            if (currX > 0)
```

```
                {
                        g.drawRect( currX, currY, =
g.getFontMetrics().stringWidth(selectedNode.reason.name) + 4, =
g.getFontMetrics().getHeight() + 4);
                        //g.drawString(selectedNode.reason.name, currX+2, =
g.getFontMetrics().getHeight() + 2);
                }
        g.dispose();
    }
=09
        public boolean mouseDrag(Event event, int x, int y)
        {
                if (selectedNode =3D=3D null)
                        return true;
                if (selectedNode.reason.id < 0)
                {
                        selectedNode.expand();
                        redraw();
                        return true;
                }
                dragNode(x, y);
                return true;
        }

        public boolean mouseUp(Event event, int x, int y)
        {
                dragNode(-1, -1);
                repaint();
                if (x > size().width)
```

```
                    postEvent(new Event(this, 10000, new Integer(100 * y / =
size().height)));
                return true;
        }

        public boolean mouseDown(Event event, int x, int y)
        {
          int index =3D (y/cellSize) + sbVPosition;
=09
          if (index > viewCount-1)
          {
                    descrip =3D null;
                    redraw();
            return false; //clicked below the last node
          }
=09
          TreeNode oldNode =3D selectedNode;
          TreeNode newNode =3D (TreeNode)v.elementAt(index);
          int newDepth =3D newNode.getDepth();
=09
          changeSelection(newNode);
          Rectangle toggleBox =3D new Rectangle(cellSize*newDepth + =
cellSize/4,
                              (index-sbVPosition)*cellSize + =
clickSize/2,
                              clickSize, clickSize);
                if (event.modifiers !=3D Event.META_MASK)
                {
                    if (toggleBox.inside(x,y))
```

```
                {
                        newNode.toggle();
                        sendActionEvent(event);
                        redraw();
                }
                else
                {
                        // check for double click
                        long currentTime =3D event.when;
        =09
                        if ((newNode=3D=3DoldNode) && =
((event.when-timeMouseDown)<doubleClickResolution))
                        {
                                newNode.toggle();
                                redraw();
                                sendActionEvent(event);
                                postEvent(new Event(this, 10000, null));
                                return false;
                        }
                        else
                        {
                                //single click action could be added here
                                timeMouseDown =3D event.when;
                        }
        =09
                }
        }
        =09
                if ((newNode !=3D null) && (event.modifiers =3D=3D
```

```
Event.META_MASK))
                {
                        descrip =3D newNode.reason.descrip;
                        redraw();
                } else {
                        descrip =3D null;
                        redraw();
                }
        =09
                return true;
        }
=09
        public boolean mouseMove(Event event, int x, int y)
        {
                return true;
        }

        public boolean keyDown(Event event, int key)
        {
          int index =3D v.indexOf(selectedNode);
=09
          switch (key)
          {
            case 10:   //enter key
              sendActionEvent(event);
              requestFocus();
              break;
            case Event.LEFT:   //left arrow
              if (selectedNode.isExpanded())
```

```
            {
               selectedNode.toggle();
               redraw();
               break;
            }
            // else drop through to "UP" with no "break;"
         case Event.UP:
            if (index > 0)
            {
               index--;
               changeSelection((TreeNode)v.elementAt(index));
               requestFocus();
            }
            break;
         case Event.RIGHT:
            if (selectedNode.isExpandable() && = (!selectedNode.isExpanded()))
            {
               selectedNode.toggle();
               sendActionEvent(event);
               redraw();
               break;
            }
=09
            if (!selectedNode.isExpandable())
            {
               break;
            }
            // else drop thru' to DOWN
         case Event.DOWN:
```

```
                if (index < viewCount-1)
                {
                  index++;
                  changeSelection((TreeNode)v.elementAt(index));
                  requestFocus();
                }
                break;
              }
=09
              return false;
            }
=09
            private void sendActionEvent(Event event)
            {
              int id =3D event.id;
              Object arg =3D event.arg;
              event.id =3D Event.ACTION_EVENT;
              event.arg =3D new String(selectedNode.getText());
              postEvent(event);
              event.id =3D id;
              event.arg =3D arg;
            }
=09
            /**
             * Returns the currently selected node.
             */
            public TreeNode getSelectedNode()
            {
              return selectedNode;
```

```
        }
=09
        /**
         * Gets the text of the currently selected node.
         * @return the text of the currently selected node or null if no node
         * is selected
         */
        public String getSelectedText()
        {
          if (selectedNode=3D=3Dnull)
          {
            return null;
          }
=09
          return selectedNode.getText();
        }
=09
        private void changeSelection(TreeNode node)
        {
          if (node =3D=3D selectedNode)
          {
            return;
          }
          TreeNode oldNode =3D selectedNode;
          selectedNode =3D node;
          drawNodeText(oldNode, (v.indexOf(oldNode) - sbVPosition)*cellSize, =
true);
          drawNodeText(node,   (v.indexOf(node)   - sbVPosition)*cellSize, =
true);
```

```
            // send select event
=09
            int index =3D v.indexOf(selectedNode);
=09
            postEvent(new Event(this, SEL_CHANGED, selectedNode));
=09
            if (index < sbVPosition)
            { //scroll up
              sbVPosition--;
              sbV.setValue(sbVPosition);
              redraw();
              return;
            }
=09
            if (index >=3D sbVPosition + (viewHeight-cellSize/2)/cellSize)
            {
              sbVPosition++;
              sbV.setValue(sbVPosition);
              redraw();
              return;
            }
=09
            repaint();
          }
=09
          public void update (Graphics g)
          {
            //(eliminates background draw to reduce flicker)
            paint(g);
```

```
        }
=09
        public void paint (Graphics g)
        {
5               Dimension d =3D size();
=09
                if ((d.width !=3D viewWidth) || (d.height !=3D viewHeight))
                        redraw();
                g.drawImage(im1, 0, 0, this);
10      }
=09
        public void redraw()
        {
          resetVector();
15=09
          if (viewCount > viewHeight/cellSize)
          {
            // need the vertical scrollbar
            sbV.setValues(sbVPosition, (viewHeight/cellSize), 0, =
20  viewCount-2);
                        sbV.setPageIncrement(1);
                        sbVWidth =3D sbV.preferredSize().width;
                        getParent().paintAll(getParent().getGraphics());
            sbV.show();
25          layout();
          }
          else
          {
            sbV.hide();
```

```
                sbVWidth =3D 1;
                sbVPosition =3D 0;
                layout();
              }
=09
              drawTree();
              repaint();
            }
=09
    /**
     * Draws the ReasonTree into an offscreen image. This is used for =
cleaner refresh.
     */
          public void drawTree()
          {
                Dimension d =3D size();
=09
                if ((d.width !=3D viewWidth) || (d.height !=3D viewHeight) || =
(g1=3D=3Dnull))
                {
              // size has changed, must resize image
                    im1 =3D createImage(d.width, d.height);
                    if (g1 !=3D null) {
                          g1.dispose();
                    }
              g1 =3D im1.getGraphics();
              viewWidth=3Dd.width;
              viewHeight=3Dd.height;
                }
```

```
=09
        Font f =3D getFont();  //unix version might not provide a default =
font
=09
            //Make certain there is a font
        if (f =3D=3D null)
        {
          f =3D new Font("TimesRoman", Font.PLAIN, 13);
          g1.setFont(f);
          setFont(f);
        }
=09
        //Make certain the graphics object has a font (Mac doesn't seem to)
            if (f !=3D null)
            {
                if (g1.getFont() =3D=3D null)
                    g1.setFont(f);
            }
      =09
        fm =3D g1.getFontMetrics();
        g1.setColor(getBackground());
        g1.fillRect(0,0,viewWidth,viewHeight);  // clear image
      =09
        //do drawing for each visible node
        int lastOne=3DsbVPosition+viewHeight/cellSize+1;
      =09
        if (lastOne > viewCount)
        {
          lastOne =3D viewCount;
```

```
        }
=09
        TreeNode outerNode =3D (TreeNode)v.elementAt(sbVPosition);
        for (int i=3DsbVPosition; i<lastOne; i++)
        {
          TreeNode node=3D(TreeNode)v.elementAt(i);
          int x =3D cellSize*(node.depth + 1);
          int y =3D (i-sbVPosition)*cellSize;
=09
          // draw lines
          g1.setColor(getForeground());
=09
          // draw vertical sibling line
          if (node.sibling !=3D null)
          {
            int k =3D v.indexOf(node.sibling) - i;
=09
            if (k > lastOne)
            {
              k =3D lastOne;
            }
=09
            drawDotLine(x - cellSize/2, y + cellSize/2,
                    x - cellSize/2, y + cellSize/2 +  k*cellSize);
=09
          }
          // if sibling is above page, draw up to top of page for this =
level
          for (int m=3D0; m<i; m++)
```

```
                {
                  TreeNode sib =3D (TreeNode)v.elementAt(m);
=09
                  if ((sib.sibling =3D=3D node) && (m<sbVPosition))
                  {
                    drawDotLine (x - cellSize/2, 0,
                            x - cellSize/2, y + cellSize/2);
                  }
                }
=09
                // draw vertical child lines
                if (node.isExpanded())
                {
                  drawDotLine(x + cellSize/2, y + cellSize -2 ,
                          x + cellSize/2, y + cellSize + cellSize/2);
                }
                // draw node horizontal line
                g1.setColor(getForeground());
                drawDotLine(x - cellSize/2, y + cellSize/2,
                          x + cellSize/2, y + cellSize/2);
=09
                // draw toggle box
                if (node.isExpandable())
                {
                  g1.setColor(getBackground());
                  g1.fillRect(cellSize*(node.depth) + cellSize/4, y + =
clickSize/2, clickSize, clickSize );
                  g1.setColor(getForeground());
                  g1.drawRect(cellSize*(node.depth) + cellSize/4, y + =
```

```
clickSize/2, clickSize, clickSize );
                // cross hair
                g1.drawLine(cellSize*(node.depth) + cellSize/4 +2,        =
  y + cellSize/2,
5                       cellSize*(node.depth) + cellSize/4 + clickSize =
-2, y + cellSize/2);
=09
                if (!(node.isExpanded()))
                {
10                 g1.drawLine(cellSize*(node.depth) + cellSize/2, y + =
clickSize/2 +2,
                           cellSize*(node.depth) + cellSize/2, y + =
clickSize/2 + clickSize -2);
                }
15          }
=09
            // draw node image
            Image nodeImage =3D node.getImage();
=09
20          if (nodeImage !=3D null)
            {
              g1.drawImage(nodeImage, x+imageInset, y, this);
            }
=09
25          // draw node text
            if (node.getText() !=3D null)
            {
              drawNodeText(node, y, node=3D=3DselectedNode);
            }
```

```
=09
            if(outerNode.depth>node.depth)
               outerNode =3D node;
            }
=09
            // draw outer vertical lines
            while((outerNode =3D outerNode.parent) !=3D null)
            {
               if(outerNode.sibling !=3D null)
                  drawDotLine (cellSize*(outerNode.depth + 1) - cellSize/2, =
0,
                        cellSize*(outerNode.depth + 1) - cellSize/2, = d.height);;
            }
         =09


            if (descrip !=3D null)
            {
                  int width =3D size().width - 20 - sbVWidth;
            =09
                  Vector lines =3D Globals.breakIntoLines(descrip, width, =
g1.getFontMetrics());
                  int y =3D size().height - (lines.size() * =
g1.getFontMetrics().getHeight());
                  g1.setColor(new Color(255, 255, 204));
                  g1.fillRect(0, y - g1.getFontMetrics().getHeight(), size().width, =
size().height);
                  g1.setColor(getForeground());
                  y +=3D g1.getFontMetrics().getHeight() >> 1;
                  for (int i =3D 0; i < lines.size(); i++)
```

```
                    {
                        g1.drawString((String)lines.elementAt(i), 10, y);
                        y +=3D g1.getFontMetrics().getHeight();
                    }
                }
                // draw border
            g1.setColor(getBackground());
                g1.setColor(getForeground());
            g1.drawRect(0,0,viewWidth - sbVWidth, viewHeight - 1);
        }
=09
        private void drawNodeText(TreeNode node, int yPosition, boolean =
eraseBackground)
        {
            Color fg, bg;
            int depth=3Dnode.depth;
            Image nodeImage =3D node.getImage();
            int textOffset =3D ((depth + 1) * (cellSize)) + cellSize + =
textInset - (nodeImage=3D=3Dnull ? 12:0);
=09
            if (node=3D=3DselectedNode)
            {
              fg=3DfgHighlightColor;
              bg=3DbgHighlightColor;
            }
            else
            {
              fg =3D getForeground();
              bg =3D getBackground();
```

```
            }
=09
            if (eraseBackground)
            {
               g1.setColor(bg);
               g1.fillRect(textOffset-1, yPosition+1, =
fm.stringWidth(node.getText())+4, cellSize-1);
            }
=09
            g1.setColor(fg);
            g1.drawString(node.getText(), textOffset, yPosition + cellSize - =
textBaseLine);
         }
=09
         private void drawDotLine(int x0, int y0, int x1, int y1)
         {
           if (y0=3D=3Dy1)
           {
              for (int i =3D x0; i<x1; i+=3D2)
              {
                g1.drawLine(i,y0, i, y1);
              }
           }
           else
           {
              for (int i =3D y0; i<y1; i+=3D2)
              {
                 g1.drawLine(x0, i, x1, i);
              }
```

```
            }
        }
=09
=09
=09
        public synchronized Dimension preferredSize()
        {
            return new Dimension(175, 125);
        }
=09
        public synchronized Dimension minimumSize()
        {
            return new Dimension(50, 50);
        }
        public void setLayout(LayoutManager lm)
        {
        }
}
class InvalidTreeNodeException
    extends Exception
{
}
```

What is claimed is: